Rate-Monotonic Schedulability of Implicit-Deadline Tasks is NP-hard Beyond Liu and Layland's Bound

Pontus Ekberg
Uppsala University
pontus.ekberg@it.uu.se

***Abstract*—We study the computational complexity of the Fixed-Priority (FP) schedulability problem for sporadic or synchronous periodic tasks with implicit deadlines on a single preemptive processor. This problem is known to be (weakly) NP-complete in the general case, but Liu and Layland's classic utilization bound trivially provides a polynomial-time solution for task sets with Rate-Monotonic (RM) priorities and utilization bounded from above by $\ln(2)$, or approximately 69%. Here we show that $\ln(2)$ is in fact the sharp boundary between computationally easy and hard schedulability testing: The FP-schedulability problem is NP-complete even if restricted to task sets with RM priorities and utilization bounded from above by any constant $c > \ln(2)$. This disproves a conjecture by Rothvoß. Further, we show that if a non-RM priority ordering can be specified, then the FP-schedulability problem is NP-complete already when utilization is bounded by any constant $c > 0$.**

I. Introduction

The *utilization bound* for Fixed-Priority (FP) scheduling is a seminal result in real-time scheduling theory, which was described by Liu and Layland [1] in the early '70s and independently by Serlin [2] around the same time. According to this bound, a set of n implicit-deadline synchronous periodic tasks executing on a single preemptive processor with Rate Monotonic (RM) priority ordering is always schedulable if the tasks' total utilization is not greater than $n(2^{1/n}-1)$. The limit of this expression as $n \to \infty$ is $\ln(2)$, or approximately 69%. The bound therefore provides a sufficient schedulability test by guaranteeing that any task set with total utilization bounded by $\ln(2)$ is schedulable.

The hyperbolic bound by Bini et al. [3] can be used to demonstrate the schedulability of more task sets without sacrificing any of the simplicity or elegance of the original bound. It does so by considering the utilization of individual tasks, but is still only sufficient for task sets with total utilization larger than $\ln(2)$. Exact schedulability analysis is instead more involved, and is typically carried out with the response-time analysis that was first described by Joseph and Pandya [4]. While it is trivial to test whether a task set's utilization is less than Liu and Layland's bound, the response-time analysis of Joseph and Pandya requires pseudo-polynomial time to run. It has been shown by Ekberg and Yi [5] that the general FP-schedulability problem for implicit-deadline tasks is (weakly) NP-complete, even when restricted to RM priorities. Pseudo-polynomial time is therefore the best we could do for this problem unless $\mathsf{P} = \mathsf{NP}$.

This work was supported by Swedish Research Council grant 2018-04446.

However, the reduction used in [5] creates task sets with utilization arbitrarily close to 1. It is natural then to ask whether the complexity of the schedulability problem would decrease if we put an upper bound on the utilization of considered task sets. Clearly this is true if we bound the utilization to be no more than $\ln(2)$, but what if utilization is bounded by some constant c that is larger than $\ln(2)$ but smaller than 1?

This question was already raised by Eisenbrand and Rothvoß [6] in their work showing that it is NP-hard to approximate response times within a constant factor under FP-scheduling. Rothvoß [7] later conjectured that if utilization is bounded by any constant $c < 1$, then response times can be calculated in polynomial time for FP-scheduling of implicit-deadline task sets with RM priority ordering (Conjecture 8.11 in [7]). This would imply, of course, that FP-schedulability could be solved in polynomial time as well.

Further evidence that bounding utilization could decrease complexity comes from schedulability tests for Earliest Deadline First (EDF) scheduling with constrained or arbitrary deadlines. There, the well-known analysis by Baruah et al. [8] requires exponential time in the general case, but runs in pseudo-polynomial time if utilization is bounded by any constant $c < 1$. Later it was shown [9], [10] that this is the best one can do unless $\mathsf{P} = \mathsf{NP}$, and hence that bounding the utilization by a constant $c < 1$ really does decrease the inherent complexity of the EDF-schedulability problem. A priori, it seems reasonable to suspect a similar effect for FP-schedulability.

Contributions: In this paper we show that for all constants c, such that $c > \ln(2)$, the FP-schedulability problem for implicit-deadline tasks with RM priorities and utilization bounded by c is (weakly) NP-complete. This means that $\ln(2)$ is the sharp transition point where bounding the utilization to any $c \leqslant \ln(2)$ results in a computationally easy (indeed, trivial) schedulability problem while bounding it by any $c > \ln(2)$ instead results in a computationally hard schedulability problem. This disproves the conjecture by Rothvoß [7] described above.

Since RM is the optimal priority ordering for task sets with implicit deadlines, a corollary is that the same hardness result also applies to the variant of the problem where we are asked if a given task set is schedulable with *any* priority ordering.

Last, in this paper we observe that if we can specify some particular non-RM priority ordering, then the schedulability problem is NP-complete already when utilization is bounded by any constant $c > 0$.

	Implicit deadlines ($d = p$)	Constrained deadlines ($d \leqslant p$)	Arbitrary deadlines (d, p unrelated)
Arbitrary utilization	Weakly NP-complete [5] Pseudo-polynomial time algorithm exists [4]	Weakly NP-complete [5] Pseudo-polynomial time algorithm exists [4]	Weakly NP-hard [5] Exponential time algorithm exists [11] (*Open*)
Utilization bounded by a constant $c < 1$	Weakly NP-complete for *(i)* $c > 0$ and arbitrary priorities, or *(ii)* $c > \ln(2)$ and RM (from this work) In P for $c \leqslant \ln(2)$ and RM [1]	Weakly NP-complete for $c > 0$ [5] Pseudo-polynomial time algorithm exists [4]	Weakly NP-hard for $c > 0$ [5] Pseudo-polynomial time algorithm exists [11] (*Open*)

Fig. 1. The complexity landscape of the fixed-priority schedulability problem for sporadic or synchronous periodic tasks on a single preemptive processor. All the hardness results reported in this table hold even if we are restricted to RM or DM priority orderings, unless otherwise specified.

Open problems: The current complexity landscape of the FP-schedulability problem for periodic tasks on a single preemptive processor is outlined in Figure 1. While complexity is now well understood with implicit or constrained deadlines in this setting, we note that the exact complexity of the schedulability problem with arbitrary deadlines is still open: Membership in NP is unknown and no pseudo-polynomial time algorithm is known for the general case even though only weak hardness has been shown for it.[1]

II. Preliminaries

Here we briefly recall some definitions and results from real-time scheduling theory that are relevant for this paper.

A synchronous[2] periodic task τ_i is represented by a triple of positive integers, $\tau_i = (C_i, D_i, T_i)$, which represent the task's worst-case execution time, relative deadline and period, respectively. Each task τ_i generates an unbounded sequence of independent jobs $[J_0^i, J_1^i, J_2^i, \ldots]$, where job J_k^i

- is released at time point kT_i,
- has deadline at time point $kT_i + D_i$, and
- must be executed for up to C_i time units between release time and deadline.

In this paper we consider scheduling on a single preemptive processor, where jobs can be preempted and later resumed at no additional cost or penalty. We say that a job is *ready* if it has been released, but has not yet executed to completion.

The *utilization* $\mathrm{U}(\tau_i)$ of a task τ_i is the fraction of the available processor time that it can require:

$$\mathrm{U}(\tau_i) \stackrel{\text{def}}{=} \frac{C_i}{T_i}. \tag{1}$$

A task set $\mathcal{T}$ is a finite (multi-)set of tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$ and its utilization $\mathrm{U}(\mathcal{T})$ is simply

$$\mathrm{U}(\mathcal{T}) \stackrel{\text{def}}{=} \sum_{\tau_i \in \mathcal{T}} \mathrm{U}(\tau_i). \tag{2}$$

The *hyper-period* $\mathrm{HP}(\mathcal{T})$ of a task set $\mathcal{T}$ is the shortest time interval after which the pattern of job releases repeats itself:

$$\mathrm{HP}(\mathcal{T}) \stackrel{\text{def}}{=} \mathrm{lcm}\{T_i \mid \tau_i \in \mathcal{T}\}. \tag{3}$$

We say that a task set $\mathcal{T}$ has

- *implicit deadlines* if $D_i = T_i$ for all $\tau_i \in \mathcal{T}$,
- *constrained deadlines* if $D_i \leqslant T_i$ for all $\tau_i \in \mathcal{T}$, and
- *arbitrary deadlines* if no restrictions are placed on the relation between D_i and T_i.

A Fixed-Priority (FP) scheduler takes a fixed total order on the tasks (the *priority ordering*) and executes the jobs in this order. That is, at every time point it would execute the ready job that was released by the task with the highest priority, preempting any job from a lower-priority task if needed. The Rate-Monotonic (RM) priority ordering assigns higher priorities to tasks with shorter periods (ties broken arbitrarily), and was shown by Liu and Layland [1] to be the optimal fixed priority ordering for task sets with implicit deadlines.

An alternative task model is offered by *sporadic* tasks. A sporadic task τ_i is represented by the same three parameters,

[1]Though it is not mentioned by Lehoczky [11], one can see that the schedulability test described in [11] for arbitrary-deadline tasks runs in exponential time in the general case and in pseudo-polynomial time with bounded utilization. The latter can be seen to hold because the longest busy period must be pseudo-polynomially bounded when utilization is bounded by a constant smaller than 1.

[2]These tasks are called *synchronous* periodic because all tasks release their first jobs at the same time point. If different initial offsets can be specified for the releases of the first jobs, then tasks are called *asynchronous* periodic instead, or sometimes just periodic.

$\tau_i = (C_i, D_i, T_i)$, with the only difference that T_i now denotes the *minimum separation* (often still called "period") between any two consecutive job releases from τ_i. Each set of sporadic tasks could therefore generate an infinite number of distinct sequences of jobs, but it is known (e.g., [11]) that all of those are FP-schedulable if and only if the single job sequence generated by the corresponding set of synchronous periodic tasks where all jobs execute for their worst-case execution time is FP-schedulable. This job sequence is called the *synchronous arrival sequence*. It follows that the FP-schedulability problems for sporadic task sets and synchronous periodic task sets are equivalent, and therefore all results reported here apply to both. In the following we write just "task" to interchangeably mean a task that is either sporadic or synchronous periodic.

The *worst-case response time* of a task is the maximum delay between a release of a job from that task and its completion time. It is known [1] that if the first job of a task finishes before the next job of that task is released in the synchronous arrival sequence, then the worst-case response time of the task is the same as the response time of the first job.

Liu and Layland [1] said that a task set *fully utilizes* the processor if it is FP-schedulable, but increasing the worst-case execution time of any task would make the task set unschedulable. They then defined the *least upper bound to processor utilization* as the infimum of the utilizations of all task sets that fully utilize the processor. Their utilization bound is stated with this terminology.

Theorem II.1 (Liu and Layland [1])**.** *The least upper bound to processor utilization (as defined above) of task sets with n implicit-deadline tasks with RM priorities is $n(2^{1/n} - 1)$.* □

We get the following corollary, which will be useful.

Corollary II.2. *For every $c > \ln(2)$, there exists an implicit-deadline task set $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with $\mathrm{U}(\mathcal{T}) < c$ that fully utilizes the processor with RM priorities.*

Proof: Let $n \in \mathbb{N}^+$ be such that $\ln(2) < n(2^{1/n} - 1) < c$. Such an n exists since $\lim_{n\to\infty} n(2^{1/n} - 1) = \ln(2)$ and $n(2^{1/n} - 1) > \ln(2)$ for all positive n. By Theorem II.1, there exists task sets $\mathcal{T}$ of n tasks with $\mathrm{U}(\mathcal{T})$ arbitrarily close to $n(2^{1/n} - 1)$ that fully utilize the processor. Such a task set $\mathcal{T}$ with $\mathrm{U}(\mathcal{T}) < c$ must then exist since $n(2^{1/n} - 1) < c$. ■

It should be noted here that Liu and Layland seem to have assumed in [1] that task parameters can be arbitrary positive real-valued numbers, while we here assume that task parameters are positive integer-valued numbers. The reason for this choice is simply that real-valued numbers do not in general have finite representations and therefore can not be used without restriction as inputs in standard models of computation. Further, we want to use integer parameters instead of rational ones because we then get the strongest hardness results (if a problem is hard with integer parameters then it must also be so with rational parameters).

Fortunately, Liu and Layland's result (Theorem II.1 above) still holds when restricted to integer-valued parameters, as noted by Devillers and Goossens [12], who also identified and revised some weaknesses in the argumentation of Liu and Layland. We therefore have Theorem II.1 (and Corollary II.2) also in this setting with integer parameters.

Last, we will show our new hardness results by reducing from the below class of FP-schedulability problems for task sets with constrained deadlines, which are known to be NP-hard.[3]

Theorem II.3 (Ekberg and Yi [5])**.** *Deciding whether a given set of sporadic or synchronous periodic tasks is FP-schedulable on a single preemptive processor is (weakly)* NP*-hard, even when restricted to task sets with*

- *utilization bounded by any constant $c > 0$,*
- *RM priority ordering,*
- *implicit deadlines for all tasks except the task with lowest priority, and*
- *a constrained deadline for the lowest-priority task.* □

III. Hardness with Rate-Monotonic Priorities

In this section we show that the FP-schedulability problem for implicit-deadline tasks is (weakly) NP-hard even when restricted to RM priorities and task sets with utilization bounded from above by any constant $c > \ln(2)$. Because there is no minimum c such that $c > \ln(2)$ we would not be satisfied by showing this only for some concrete value of c, no matter how close it is to $\ln(2)$. Instead, we will here show how to construct an appropriate polynomial-time many-one reduction for any such c. The resulting *family of reductions* will demonstrate that for all $c > \ln(2)$, there exists a reduction that shows that the corresponding FP-schedulability problem is NP-hard.

A. Selecting some constants and a source problem

Let the constant $c > \ln(2)$ that bounds utilization for our target problem be given. We assume, without loss of generality, that $c \leqslant 1$. We will define some additional constants before selecting the exact source problem for the reduction. First, let

$$c_{\text{FIX}} \stackrel{\text{def}}{=} \frac{c + \ln(2)}{2} \tag{4}$$

and note that $\ln(2) < c_{\text{FIX}} < c$. Recall that Liu and Layland [1] said that a task set *fully utilizes* the processor if it is FP-schedulable, but would cease to be schedulable if the worst-case execution time of any task was increased. We say that such a task set is *minimal* if it would no longer fully utilize the processor if any task was removed from it.

Now, let $\mathcal{T}_{\text{FIX}}$ denote a *fixed* (constant) task set with the following four properties.

[3] Theorem II.3 here differs slightly from the original statement (part *(ii)* of Theorem II.8 in [5]). In [5], DM priority ordering is specified instead of RM, and it is not specified that higher-priority tasks can be restricted to implicit deadlines. However, we note that it follows immediately from that reduction (Eqs. (20)–(28) in [5]) that the DM and RM priority orderings are the same for the task sets produced, and that all the tasks except the lowest-priority one do in fact have implicit deadlines. We conclude that the variant stated here follows directly from the results in [5].

(i) $\mathrm{U}(\mathcal{T}_{\text{FIX}}) \leqslant c_{\text{FIX}}$,

(ii) $\mathcal{T}_{\text{FIX}}$ has implicit deadlines,

(iii) $\mathcal{T}_{\text{FIX}}$ fully utilizes the processor (with RM priorities), and

(iv) $\mathcal{T}_{\text{FIX}}$ is minimal.

We observe that such a task set exists.

Lemma III.1. *A task set $\mathcal{T}_{\text{FIX}}$ satisfying criteria (i)–(iv) exists.*

Proof: Because $c_{\text{FIX}} > \ln(2)$ we know from Corollary II.2 that there exists an implicit-deadline task set $\mathcal{T}_{\text{FIX}}$ that fully utilizes the processor with $\mathrm{U}(\mathcal{T}_{\text{FIX}}) \leqslant c_{\text{FIX}}$. If $\mathcal{T}_{\text{FIX}}$ is not minimal, we can make it so by repeatedly removing some suitable task from $\mathcal{T}_{\text{FIX}}$ until it becomes minimal. ∎

The task set $\mathcal{T}_{\text{FIX}}$ will be used for the reduction described in Section III-B. We can let $\mathcal{T}_{\text{FIX}}$ be any task set that satisfies criteria *(i)–(iv)* above, but we will keep this $\mathcal{T}_{\text{FIX}}$ fixed for the reduction. In other words, given only constant c_{FIX} we pick a $\mathcal{T}_{\text{FIX}}$ that will be hard-coded into the reduction algorithm, meaning that $\mathcal{T}_{\text{FIX}}$ will be entirely independent of the inputs to the reduction. From the point of view of the reduction, the number of tasks in $\mathcal{T}_{\text{FIX}}$ is therefore a constant, as are all the parameter values of the tasks in $\mathcal{T}_{\text{FIX}}$.

Let $\mathcal{T}_{\text{FIX}}$ be fixed and the following notation defined.

- $\tau_{\text{FIX}}^{\text{lp}}$ is the lowest-priority task in $\mathcal{T}_{\text{FIX}}$ (under RM),
- $C_{\text{FIX}}^{\text{lp}}$ and $T_{\text{FIX}}^{\text{lp}}$ are the worst-case execution time and period of the task $\tau_{\text{FIX}}^{\text{lp}}$, respectively,
- $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ is the set of all higher-priority tasks in $\mathcal{T}_{\text{FIX}}$, and
- Δ is the worst-case response time of $\tau_{\text{FIX}}^{\text{lp}}$ under RM scheduling of $\mathcal{T}_{\text{FIX}}$.

Now, let

$$c_{\text{IN}} \stackrel{\text{def}}{=} \min\left(c - c_{\text{FIX}}, \frac{1}{2\Delta}\right). \tag{5}$$

We note that $c_{\text{IN}} > 0$ and since Δ is a constant c_{IN} is also a constant. This leads us, finally, to the exact source problem of the reduction.

Outline: We will describe a polynomial-time many-one reduction from the FP-schedulability problem that is described in Theorem II.3, where utilization is bounded by c_{IN}. Given any instance of this problem—some input task set $\mathcal{T}_{\text{IN}}$—the reduction will produce an output task set $\mathcal{T}_{\text{OUT}}$ with implicit deadlines, such that $\mathrm{U}(\mathcal{T}_{\text{OUT}}) \leqslant c$ and $\mathcal{T}_{\text{OUT}}$ is FP-schedulable with RM priorities if and only if $\mathcal{T}_{\text{IN}}$ is.

B. Producing $\mathcal{T}_{\text{OUT}}$ from $\mathcal{T}_{\text{IN}}$

We have picked a source problem for our reduction, namely the FP-schedulability problem described in Theorem II.3 for task sets with utilization bounded by c_{IN}. We will now describe the reduction by showing how any instance $\mathcal{T}_{\text{IN}}$ of this source problem can be transformed in polynomial time into an instance $\mathcal{T}_{\text{OUT}}$ of our target problem. In the next section we will see that FP-schedulability is preserved by this transformation.

Let $\mathcal{T}_{\text{IN}}$ be given and the following notation defined.

- $\tau_{\text{IN}}^{\text{lp}}$ is the lowest-priority task in $\mathcal{T}_{\text{IN}}$ (under RM),
- $C_{\text{IN}}^{\text{lp}}$, $D_{\text{IN}}^{\text{lp}}$ and $T_{\text{IN}}^{\text{lp}}$ are the worst-case execution time, relative deadline and period of task $\tau_{\text{IN}}^{\text{lp}}$, respectively, and
- $\mathcal{T}_{\text{IN}}^{\text{hp}}$ is the set of all higher-priority tasks in $\mathcal{T}_{\text{IN}}$.

Note that by definition, all tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ have implicit deadlines and $\tau_{\text{IN}}^{\text{lp}}$ has a constrained deadline. We assume, without loss of generality, that $C_{\text{IN}}^{\text{lp}} \leqslant D_{\text{IN}}^{\text{lp}}$.

The reduction will depend heavily on the constant task set $\mathcal{T}_{\text{FIX}}$, which was described in the previous section. The task set $\mathcal{T}_{\text{OUT}}$ will be the result of merging $\mathcal{T}_{\text{IN}}$ with $\mathcal{T}_{\text{FIX}}$ as described below. First we scale all the higher-priority tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ and $\mathcal{T}_{\text{FIX}}^{\text{hp}}$. The tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ will be uniformly scaled by Δ, and we denote the resulting task set $\mathcal{T}_{\text{IN}}^{\Delta}$:

$$\mathcal{T}_{\text{IN}}^{\Delta} \stackrel{\text{def}}{=} \{(\Delta C_i, \Delta D_i, \Delta T_i) \mid (C_i, D_i, T_i) \in \mathcal{T}_{\text{IN}}^{\text{hp}}\}. \tag{6}$$

The tasks in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ will instead be uniformly scaled by a number k, resulting in $\mathcal{T}_{\text{FIX}}^{k}$:

$$\mathcal{T}_{\text{FIX}}^{k} \stackrel{\text{def}}{=} \{(kC_i, kD_i, kT_i) \mid (C_i, D_i, T_i) \in \mathcal{T}_{\text{FIX}}^{\text{hp}}\}, \tag{7}$$

where

$$k \stackrel{\text{def}}{=} 2\Delta\mathrm{HP}(\mathcal{T}_{\text{IN}}) + D_{\text{IN}}^{\text{lp}}. \tag{8}$$

Together, these will form the higher-priority tasks $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ in the output task set $\mathcal{T}_{\text{OUT}}$.

$$\mathcal{T}_{\text{OUT}}^{\text{hp}} \stackrel{\text{def}}{=} \mathcal{T}_{\text{IN}}^{\Delta} \cup \mathcal{T}_{\text{FIX}}^{k} \tag{9}$$

The final piece is the low-priority task $\tau_{\text{OUT}}^{\text{lp}}$ with execution time

$$C_{\text{OUT}}^{\text{lp}} \stackrel{\text{def}}{=} kC_{\text{FIX}}^{\text{lp}} + \Delta(C_{\text{IN}}^{\text{lp}} - D_{\text{IN}}^{\text{lp}}) - 2\Delta^2\mathrm{HP}(\mathcal{T}_{\text{IN}})\mathrm{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}) \tag{10}$$

and with relative deadline and period

$$D_{\text{OUT}}^{\text{lp}} \stackrel{\text{def}}{=} T_{\text{OUT}}^{\text{lp}} \stackrel{\text{def}}{=} kT_{\text{FIX}}^{\text{lp}}. \tag{11}$$

The complete output task set $\mathcal{T}_{\text{OUT}}$ is then

$$\mathcal{T}_{\text{OUT}} \stackrel{\text{def}}{=} \mathcal{T}_{\text{OUT}}^{\text{hp}} \cup \{\tau_{\text{OUT}}^{\text{lp}}\}. \tag{12}$$

We note in the following three lemmas that $\mathcal{T}_{\text{OUT}}$ is a valid instance of the target problem: an implicit-deadline task set with positive integer task parameters and utilization bounded by c. That the reduction preserves FP-schedulability is shown in the next section.

Lemma III.2. *$\mathcal{T}_{\text{OUT}}$ has positive integer task parameters.*

Proof: The tasks in $\mathcal{T}_{\text{IN}}$ and $\mathcal{T}_{\text{FIX}}$ have positive integer parameters by definition, so the tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ must also have such parameters by Eqs. (6)–(9).

The same is true of the period (and deadline) of the low-priority task $\tau_{\text{OUT}}^{\text{lp}}$ by Eq. (11). What remains to be shown is that $C_{\text{OUT}}^{\text{lp}}$ is a positive integer. First, using Eqs. (5), (8) and

(10) we see that $C^{\text{lp}}_{\text{OUT}}$ is positive since

$$\begin{aligned} C^{\text{lp}}_{\text{OUT}} &= kC^{\text{lp}}_{\text{FIX}} + \Delta(C^{\text{lp}}_{\text{IN}} - D^{\text{lp}}_{\text{IN}}) - 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) && (13)\\ &\geqslant k - \Delta D^{\text{lp}}_{\text{IN}} - 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) && (14)\\ &\geqslant k - \Delta D^{\text{lp}}_{\text{IN}} - 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})c_{\text{IN}} && (15)\\ &\geqslant k - \Delta D^{\text{lp}}_{\text{IN}} - \Delta\text{HP}(\mathcal{T}_{\text{IN}}) && (16)\\ &\geqslant k - 2\Delta\text{HP}(\mathcal{T}_{\text{IN}}) && (17)\\ &> 0. && (18)\end{aligned}$$

Note that $\text{HP}(\mathcal{T})\text{U}(\mathcal{T})$ is an integer for any task set $\mathcal{T}$. Since $\text{HP}(\mathcal{T}^{\text{hp}}_{\text{IN}})$ divides $\text{HP}(\mathcal{T}_{\text{IN}})$, it follows that $\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}})$ must be an integer, and so is then $C^{\text{lp}}_{\text{OUT}}$ by Eq. (10). ■

Lemma III.3. $\text{U}(\mathcal{T}_{\text{OUT}}) \leqslant c$.

Proof: Note that $(C^{\text{lp}}_{\text{IN}} - D^{\text{lp}}_{\text{IN}}) \leqslant 0$, so by Eq. (10) we have

$$C^{\text{lp}}_{\text{OUT}} \;\leqslant\; kC^{\text{lp}}_{\text{FIX}}. \tag{19}$$

It follows that

$$\text{U}(\tau^{\text{lp}}_{\text{OUT}}) \;=\; \frac{C^{\text{lp}}_{\text{OUT}}}{T^{\text{lp}}_{\text{OUT}}} \;=\; \frac{C^{\text{lp}}_{\text{OUT}}}{kT^{\text{lp}}_{\text{FIX}}} \;\leqslant\; \frac{kC^{\text{lp}}_{\text{FIX}}}{kT^{\text{lp}}_{\text{FIX}}} \;=\; \text{U}(\tau^{\text{lp}}_{\text{FIX}}) \tag{20}$$

and therefore

$$\begin{aligned} \text{U}(\mathcal{T}_{\text{OUT}}) &= \text{U}(\mathcal{T}^{\Delta}_{\text{IN}}) + \text{U}(\mathcal{T}^{k}_{\text{FIX}}) + \text{U}(\tau^{\text{lp}}_{\text{OUT}}) && (21)\\ &= \text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) + \text{U}(\mathcal{T}^{\text{hp}}_{\text{FIX}}) + \text{U}(\tau^{\text{lp}}_{\text{OUT}}) && (22)\\ &\leqslant \text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) + \text{U}(\mathcal{T}^{\text{hp}}_{\text{FIX}}) + \text{U}(\tau^{\text{lp}}_{\text{FIX}}) && (23)\\ &= \text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) + \text{U}(\mathcal{T}_{\text{FIX}}) && (24)\\ &\leqslant c_{\text{IN}} + c_{\text{FIX}} && (25)\\ &\leqslant c, && (26)\end{aligned}$$

which is what we wanted to show. ■

Lemma III.4. $\mathcal{T}_{\text{OUT}}$ *has implicit deadlines.*

Proof: The task set $\mathcal{T}_{\text{OUT}}$ contains only scaled variants of the tasks in $\mathcal{T}^{\text{hp}}_{\text{IN}}$ and $\mathcal{T}^{\text{hp}}_{\text{FIX}}$, as well as the task $\tau^{\text{lp}}_{\text{OUT}}$. All the tasks in $\mathcal{T}^{\text{hp}}_{\text{IN}}$ and $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ have implicit deadlines and this does not change when task parameters are uniformly scaled. The task $\tau^{\text{lp}}_{\text{OUT}}$ has an implicit deadline by construction. ■

C. Correctness of the reduction

We know from Lemmas III.2–III.4 that $\mathcal{T}_{\text{OUT}}$ is a valid instance of the target problem. From Eqs. (6)–(12) it is clear that $\mathcal{T}_{\text{OUT}}$ can be produced from $\mathcal{T}_{\text{IN}}$ in polynomial time.[4] What remains to be shown here is that the reduction is correct: that $\mathcal{T}_{\text{OUT}}$ is FP-schedulable with RM priorities if and only if $\mathcal{T}_{\text{IN}}$ is also FP-schedulable with RM priorities.

We start with an overview. The general idea is that the scheduling of the first job of $\tau^{\text{lp}}_{\text{IN}}$ in $\mathcal{T}_{\text{IN}}$ will be "simulated" by the scheduling of the first job of $\tau^{\text{lp}}_{\text{OUT}}$ in $\mathcal{T}_{\text{OUT}}$. Since $\tau^{\text{lp}}_{\text{IN}}$ has a constrained deadline and we are only allowed to use implicit deadline tasks for $\mathcal{T}_{\text{OUT}}$, this requires us to carefully set up the interference of higher-priority tasks to recreate the effect of the constrained deadline while keeping utilization low.

[4]Since the entire task set $\mathcal{T}_{\text{FIX}}$ is constant, any properties of it, such as the worst-case response-time Δ of its lowest-priority task, can be pre-computed.

As we will see in the remainder of this section, the interference of the higher-priority tasks $\mathcal{T}^{\text{hp}}_{\text{OUT}}$ is set up so that the interval $[\Delta(k - D^{\text{lp}}_{\text{IN}}), \Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$ is a scaled replicate of the interval $[0, D^{\text{lp}}_{\text{IN}})$ in the synchronous arrival sequence of $\mathcal{T}_{\text{IN}}$.

We create the effect of a "deadline" for $\tau^{\text{lp}}_{\text{OUT}}$ at time point Δk in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$ by fully occupying the processor with higher-priority tasks from Δk until $T^{\text{lp}}_{\text{OUT}}$. Task $\tau^{\text{lp}}_{\text{OUT}}$ will execute for exactly $C^{\text{lp}}_{\text{OUT}} - \Delta C^{\text{lp}}_{\text{IN}}$ time units before time point $\Delta(k - D^{\text{lp}}_{\text{IN}})$. Therefore, $\tau^{\text{lp}}_{\text{OUT}}$ will finish by its deadline if and only if it can execute for the remaining $\Delta C^{\text{lp}}_{\text{IN}}$ time units inside the interval $[\Delta(k - D^{\text{lp}}_{\text{IN}}), \Delta k)$. No jobs from $\mathcal{T}^{k}_{\text{FIX}}$ will be active in that interval and the start of the interval aligns with a hyper-period of $\mathcal{T}^{\Delta}_{\text{IN}}$. Hence, the interference suffered by $\tau^{\text{lp}}_{\text{OUT}}$ in $[\Delta(k - D^{\text{lp}}_{\text{IN}}), \Delta k)$ is the same as the interference suffered by $\tau^{\text{lp}}_{\text{IN}}$ during its first scheduling window $[0, D^{\text{lp}}_{\text{IN}})$, but scaled by Δ. Figure 2 illustrates this overview.

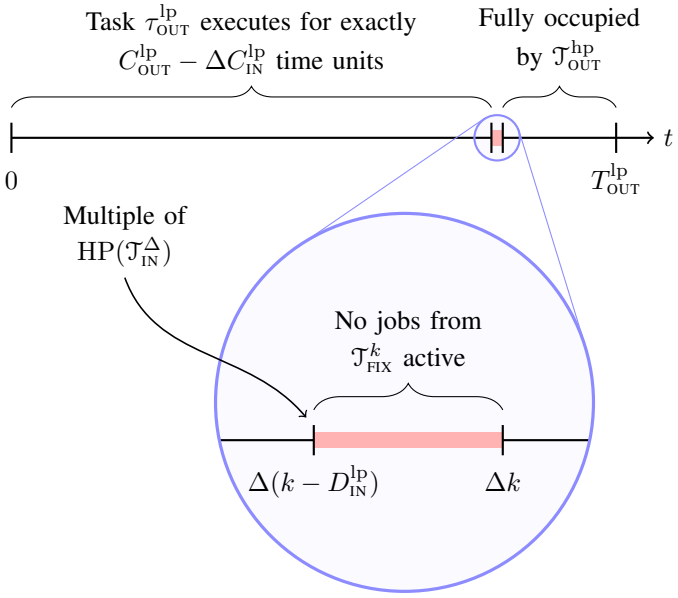

Fig. 2. The execution of $\tau^{\text{lp}}_{\text{IN}}$ in the synchronous arrival sequence of $\mathcal{T}_{\text{IN}}$ is "simulated" by $\tau^{\text{lp}}_{\text{OUT}}$ during the interval $[\Delta(k - D^{\text{lp}}_{\text{IN}}), \Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$.

We will need to establish several lemmas. We start by observing that the schedulability of $\mathcal{T}_{\text{IN}}$ hinges only on its lowest-priority task.

Lemma III.5. *If $\mathcal{T}_{\text{IN}}$ is FP-scheduled with RM priorities, then all the higher-priority tasks in $\mathcal{T}^{\text{hp}}_{\text{IN}}$ are schedulable.*

Proof: By definition, the tasks in $\mathcal{T}^{\text{hp}}_{\text{IN}}$ have implicit deadlines and $\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) < \text{U}(\mathcal{T}_{\text{IN}}) \leqslant c_{\text{IN}}$. By Eq. (5) we have $c_{\text{IN}} \leqslant 1/2 < \ln(2)$, so $\mathcal{T}^{\text{hp}}_{\text{IN}}$ is schedulable by Theorem II.1. ■

We then take a look at the priority ordering of the tasks in $\mathcal{T}_{\text{OUT}}$. Recall that $\mathcal{T}_{\text{OUT}} = \mathcal{T}^{\Delta}_{\text{IN}} \cup \mathcal{T}^{k}_{\text{FIX}} \cup \{\tau^{\text{lp}}_{\text{OUT}}\}$.

Lemma III.6. *When using RM priority ordering for $\mathcal{T}_{\text{OUT}}$, the task $\tau_{\text{OUT}}^{\text{lp}}$ has the lowest priority, and the tasks in $\mathcal{T}_{\text{FIX}}^{k}$ all have lower priorities than the tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$.*

Proof: Because of the scaling of task parameters in Eqs. (6) and (7), no period of any task in $\mathcal{T}_{\text{FIX}}^{k}$ can be less than k and no period of any task in $\mathcal{T}_{\text{IN}}^{\Delta}$ can be larger than $\Delta\text{HP}(\mathcal{T}_{\text{IN}})$. But $k > \Delta\text{HP}(\mathcal{T}_{\text{IN}})$, so all tasks in $\mathcal{T}_{\text{FIX}}^{k}$ have larger periods, and therefore lower priorities, than the tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$.

The task $\tau_{\text{OUT}}^{\text{lp}}$ has period $T_{\text{OUT}}^{\text{lp}} = kT_{\text{FIX}}^{\text{lp}}$. No task in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ has a period larger than $T_{\text{FIX}}^{\text{lp}}$ since $\tau_{\text{FIX}}^{\text{lp}}$ has the lowest priority in $\mathcal{T}_{\text{FIX}}$ under RM, so then no task in $\mathcal{T}_{\text{FIX}}^{k}$ can have a period larger than $kT_{\text{FIX}}^{\text{lp}}$. The task $\tau_{\text{OUT}}^{\text{lp}}$ can therefore be given the lowest priority in $\mathcal{T}_{\text{OUT}}$ under RM. ■

We immediately see that all the tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$ are schedulable.

Lemma III.7. *If $\mathcal{T}_{\text{OUT}}$ is FP-scheduled with RM priorities, then the tasks in the subset $\mathcal{T}_{\text{IN}}^{\Delta}$ are schedulable.*

Proof: From Lemma III.6 we know that the tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$ have the highest priorities of all tasks in $\mathcal{T}_{\text{OUT}}$. By Lemma III.5, these tasks were schedulable before the uniform scaling of their parameters in Eq. (6), so they must remain so also after the scaling. ■

As was mentioned above, the reduction works by "simulating" the scheduling of the first job of $\tau_{\text{IN}}^{\text{lp}}$ in the schedule of $\mathcal{T}_{\text{OUT}}$. This will occur during the time interval $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$. The next lemma is about the behavior of $\mathcal{T}_{\text{IN}}^{\Delta}$ before the start of this interval.

Lemma III.8. *Before time point $t = \Delta(k - D_{\text{IN}}^{\text{lp}})$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$, the subset of tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$ will together have released jobs with a total execution time of*

$$2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}), \tag{27}$$

and all those jobs will have finished no later than t.

Proof: Because $\text{HP}(\mathcal{T}_{\text{IN}}^{\text{hp}})$ divides $\text{HP}(\mathcal{T}_{\text{IN}})$, we must also have that $\text{HP}(\mathcal{T}_{\text{IN}}^{\Delta})$ divides $\Delta\text{HP}(\mathcal{T}_{\text{IN}})$. By Eq. (8) we have

$$t \;=\; \Delta(k - D_{\text{IN}}^{\text{lp}}) \;=\; 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}}), \tag{28}$$

which is a multiple of $\Delta\text{HP}(\mathcal{T}_{\text{IN}})$. It follows that $\text{HP}(\mathcal{T}_{\text{IN}}^{\Delta})$ divides t, and therefore the total execution time of all jobs released by $\mathcal{T}_{\text{IN}}^{\Delta}$ before t is

$$\begin{aligned} t\text{U}(\mathcal{T}_{\text{IN}}^{\Delta}) \;&=\; \Delta(k - D_{\text{IN}}^{\text{lp}})\text{U}(\mathcal{T}_{\text{IN}}^{\Delta}) && (29)\\ &=\; \Delta(k - D_{\text{IN}}^{\text{lp}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}) && (30)\\ &=\; 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}). && (31) \end{aligned}$$

Further, since $\text{HP}(\mathcal{T}_{\text{IN}}^{\Delta})$ divides t and $\mathcal{T}_{\text{IN}}^{\Delta}$ has implicit deadlines, all tasks of $\mathcal{T}_{\text{IN}}^{\Delta}$ have a deadline at t. Because the tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$ are schedulable by Lemma III.7, all jobs released before t must have finished no later than t. ■

Before considering the subset $\mathcal{T}_{\text{FIX}}^{k}$ we look closer at the constant task set $\mathcal{T}_{\text{FIX}}$ on which it is based. The next three lemmas describe some useful properties of $\mathcal{T}_{\text{FIX}}$.

Lemma III.9. *If $\mathcal{T}_{\text{FIX}}$ is FP-scheduled with RM priorities, then no task in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ is active in the time interval $[\Delta - 1, \Delta)$ in the synchronous arrival sequence.*

Proof: By definition, Δ is the finishing time of the first job of the lowest-priority task $\tau_{\text{FIX}}^{\text{lp}}$ in the synchronous arrival sequence of $\mathcal{T}_{\text{FIX}}$. Since task parameters are integer it follows that $\tau_{\text{FIX}}^{\text{lp}}$ executes in $[\Delta - 1, \Delta)$, and therefore none of the higher-priority tasks in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ can be active there. ■

Lemma III.10. *If $\mathcal{T}_{\text{FIX}}$ is FP-scheduled with RM priorities, then the time interval $[\Delta, T_{\text{FIX}}^{\text{lp}})$ is fully occupied by execution of tasks in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ in the synchronous arrival sequence.*

Proof: Assume that the time interval $[\Delta, T_{\text{FIX}}^{\text{lp}})$ is not fully occupied by tasks in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$. Then the processor would be idle at some points during that interval since the first job of $\tau_{\text{FIX}}^{\text{lp}}$ finished already at Δ. But if there is idle time during the interval $[0, T_{\text{FIX}}^{\text{lp}})$, then the execution time of $\tau_{\text{FIX}}^{\text{lp}}$ could be increased without it becoming unschedulable. This is a contradiction to the fact that $\mathcal{T}_{\text{FIX}}$ fully utilizes the processor. ■

Lemma III.11. *If $\mathcal{T}_{\text{FIX}}$ is FP-scheduled with RM priorities, then the first job of any task $\tau_i \in \mathcal{T}_{\text{FIX}}^{\text{hp}}$ would meet its deadline in the synchronous arrival sequence, even if that job suffered 1 additional time unit of interference from higher-priority tasks.*

Proof: Assume that the first job of some task $\tau_i \in \mathcal{T}_{\text{FIX}}^{\text{hp}}$ would miss its deadline if it suffered 1 additional time unit of interference. Then τ_i would not be schedulable if any of the tasks with higher priority than τ_i had its execution time increased, since that would cause at least 1 time unit of extra interference for the first job of τ_i. Similarly, the execution time of τ_i itself could not be increased, even by 1, without the first job missing its deadline.

But if τ_i would cease to be schedulable if any task with higher priority, or τ_i itself, has its execution time increased, then $\mathcal{T}_{\text{FIX}}$ would fully utilize the processor even if all tasks with lower priority than τ_i were removed from it. This is in contradiction to the fact that $\mathcal{T}_{\text{FIX}}$ is minimal by definition and does not fully utilize the processor if any task is removed. ■

We can now consider the behavior of $\mathcal{T}_{\text{FIX}}^{k}$ before the start of the interval $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$.

Lemma III.12. *Any job from a task in $\mathcal{T}_{\text{FIX}}^{k}$ that is released before time point $t = \Delta(k - D_{\text{IN}}^{\text{lp}})$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$ will finish no later than t.*

Proof: Consider for the moment the schedule we would get by only scheduling the tasks in subset $\mathcal{T}_{\text{FIX}}^{k}$, without the other tasks in $\mathcal{T}_{\text{OUT}}$. Since these tasks are copies of the tasks in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$, uniformly scaled by k, the schedule of these tasks would look the same as the schedule of $\mathcal{T}_{\text{FIX}}^{\text{hp}}$, only stretched so that every time unit becomes k time units long. Lemma III.9 then tells us that in this schedule of $\mathcal{T}_{\text{FIX}}^{k}$, no job is active in $[k(\Delta - 1), k\Delta)$. Since $t \in [k(\Delta - 1), k\Delta)$, all tasks released before time point t would be finished by $k(\Delta - 1)$, and no new jobs released in $[k(\Delta - 1), t)$.

However, when scheduling the whole of $\mathcal{T}_{\text{OUT}}$ the tasks in $\mathcal{T}^k_{\text{FIX}}$ suffer additional interference from the tasks in $\mathcal{T}^{\Delta}_{\text{IN}}$, which have higher priority by Lemma III.6. From Lemma III.8 we know that the tasks in $\mathcal{T}^{\Delta}_{\text{IN}}$ together will execute for $2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}})$ time units in total until time point t. Using Eqs. (5) and (8) we have

$$\begin{aligned} 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) &\leqslant 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}) && (32)\\ &\leqslant 2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})c_{\text{IN}} && (33)\\ &\leqslant \frac{2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})}{2\Delta} && (34)\\ &= \Delta\text{HP}(\mathcal{T}_{\text{IN}}) && (35)\\ &\leqslant \frac{k}{2}, && (36)\end{aligned}$$

As any job from tasks in $\mathcal{T}^k_{\text{FIX}}$ released before t would have finished before $k(\Delta-1)$ without the interference from $\mathcal{T}^{\Delta}_{\text{IN}}$, the latest completion time for such a job if we include interference from $\mathcal{T}^{\Delta}_{\text{IN}}$ is no later than

$$\begin{aligned} k(\Delta-1)+2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) &\leqslant k(\Delta-1)+\frac{k}{2} && (37)\\ &= k\Delta-\frac{k}{2} && (38)\\ &\leqslant k\Delta-\Delta\text{HP}(\mathcal{T}_{\text{IN}}) && (39)\\ &\leqslant k\Delta-\Delta T^{\text{lp}}_{\text{IN}} && (40)\\ &\leqslant \Delta(k-D^{\text{lp}}_{\text{IN}}) && (41)\\ &= t. && (42)\end{aligned}$$

It follows that if a job is released by a task in $\mathcal{T}^k_{\text{FIX}}$ before t, then it will finish no later than t even under interference from the tasks in $\mathcal{T}^{\Delta}_{\text{IN}}$. ■

We can now see that the tasks in $\mathcal{T}^k_{\text{FIX}}$ are also schedulable, and hence that the schedulability of $\mathcal{T}_{\text{OUT}}$, like $\mathcal{T}_{\text{IN}}$, hinges only on its lowest-priority task.

Lemma III.13. *If $\mathcal{T}_{\text{OUT}}$ is FP-scheduled with RM priorities, then the tasks in the subset $\mathcal{T}^k_{\text{FIX}}$ are schedulable.*

Proof: From Lemma III.11 we know that in the synchronous arrival sequence of the constant task set $\mathcal{T}_{\text{FIX}}$, the first job of any task in $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ would still meet its deadline even if it suffered additional interference of 1 time unit from higher-priority tasks. If we consider the scheduling of the tasks in the subset $\mathcal{T}^k_{\text{FIX}}$ in isolation (without the other tasks in $\mathcal{T}_{\text{OUT}}$), it follows from those tasks' uniform scaling by k that the first job of any task in $\mathcal{T}^k_{\text{FIX}}$ would still meet its deadline even if it suffered additional interference of k time units from higher-priority tasks.

If we now consider the scheduling of the whole task set $\mathcal{T}_{\text{OUT}}$, the tasks in $\mathcal{T}^k_{\text{FIX}}$ will suffer additional interference from the tasks in $\mathcal{T}^{\Delta}_{\text{IN}}$. From Lemma III.12 we know that the first job of any task in $\mathcal{T}^k_{\text{FIX}}$ will finish no later than time point $t=\Delta(k-D^{\text{lp}}_{\text{IN}})$, and from Lemma III.8 we know that the tasks in $\mathcal{T}^{\Delta}_{\text{IN}}$ will execute for a total of $2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}})$ time units before t. From Eqs. (32)–(36) we have

$$2\Delta^2\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}^{\text{hp}}_{\text{IN}}) \leqslant \frac{k}{2} \qquad (43)$$

and therefore the first job of any task in $\mathcal{T}^k_{\text{FIX}}$ will only suffer interference of up to $k/2$ time units from tasks in $\mathcal{T}^{\Delta}_{\text{IN}}$. The first job of any task in $\mathcal{T}^k_{\text{FIX}}$ then meets its deadline in the synchronous arrival sequence despite the interference from $\mathcal{T}^{\Delta}_{\text{IN}}$. We conclude that the task is schedulable since the first job has the maximum response time. ■

We can now see that the tasks in $\mathcal{T}^k_{\text{FIX}}$ are not causing any interference during time interval $[\Delta(k-D^{\text{lp}}_{\text{IN}}),\Delta k)$ where we will "simulate" $\tau^{\text{lp}}_{\text{IN}}$'s first job.

Lemma III.14. *No job from subset $\mathcal{T}^k_{\text{FIX}}$ is active during time interval $[\Delta(k-D^{\text{lp}}_{\text{IN}}),\Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$.*

Proof: From Lemma III.9 we know that no job from $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ would be released during $[\Delta-1,\Delta)$ in the synchronous arrival sequence of the constant task set $\mathcal{T}_{\text{FIX}}$. Since $\mathcal{T}^k_{\text{FIX}}$ is identical to $\mathcal{T}^{\text{hp}}_{\text{FIX}}$, except that task parameters are uniformly scaled by k, it follows that no jobs from tasks in $\mathcal{T}^k_{\text{FIX}}$ are released in interval $[k(\Delta-1),k\Delta)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$.

From Lemma III.12 we know that all jobs from subset $\mathcal{T}^k_{\text{FIX}}$ that are released before $t=\Delta(k-D^{\text{lp}}_{\text{IN}})$ are finished no later than t. Because $t>k(\Delta-1)$, it follows that no job from $\mathcal{T}^k_{\text{FIX}}$ is active during $[t,k\Delta)=[\Delta(k-D^{\text{lp}}_{\text{IN}}),\Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$. ■

The next lemma quantifies the interference of tasks in $\mathcal{T}^k_{\text{FIX}}$ before the time interval $[\Delta(k-D^{\text{lp}}_{\text{IN}}),\Delta k)$.

Lemma III.15. *In the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$, the tasks in subset $\mathcal{T}^k_{\text{FIX}}$ execute for a total duration of*

$$k(\Delta-C^{\text{lp}}_{\text{FIX}}) \qquad (44)$$

time units until time point $t=\Delta(k-D^{\text{lp}}_{\text{IN}})$.

Proof: Consider first the constant task set $\mathcal{T}_{\text{FIX}}$. Because Δ is the worst-case response time of the lowest-priority task $\tau^{\text{lp}}_{\text{FIX}}$ we know that in the synchronous arrival sequence of $\mathcal{T}_{\text{FIX}}$, the higher-priority tasks in $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ would execute for a total amount of $\Delta-C^{\text{lp}}_{\text{FIX}}$ during interval $[0,\Delta)$. Further, all jobs released by tasks in $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ before Δ would have finished before Δ, since otherwise $\tau^{\text{lp}}_{\text{FIX}}$ would not be able to execute and finish at that point. It follows that the total execution time of all jobs released by tasks in $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ before time point Δ is $\Delta-C^{\text{lp}}_{\text{FIX}}$.

Since $\mathcal{T}^k_{\text{FIX}}$ is uniformly scaled by k from $\mathcal{T}^{\text{hp}}_{\text{FIX}}$ it follows that the total execution time of all jobs released by tasks in $\mathcal{T}^k_{\text{FIX}}$ before time point Δk must be $k(\Delta-C^{\text{lp}}_{\text{FIX}})$. By Lemma III.14 we know that no jobs from tasks in $\mathcal{T}^k_{\text{FIX}}$ are active in the interval $[t,\Delta k)$, so the tasks in $\mathcal{T}^k_{\text{FIX}}$ must have executed for a total duration of $k(\Delta-C^{\text{lp}}_{\text{FIX}})$ until time point t. ■

Finally we can put the pieces together and show that the reduction preserves schedulability.

Lemma III.16. *$\mathcal{T}_{\text{OUT}}$ is FP-schedulable with RM priorities if and only if $\mathcal{T}_{\text{IN}}$ is.*

Proof: From Lemmas III.5, III.7 and III.13 we know that the task sets $\mathcal{T}_{\text{IN}}$ and $\mathcal{T}_{\text{OUT}}$ are schedulable if and only if their respective low-priority tasks are schedulable. What we need to show is therefore that $\tau_{\text{OUT}}^{\text{lp}}$ is schedulable if and only if $\tau_{\text{IN}}^{\text{lp}}$ is. Since any task with a constrained (or implicit) deadline is FP-schedulable if and only if its first job in the synchronous arrival sequence meets its deadline, we will focus on the first jobs from $\tau_{\text{IN}}^{\text{lp}}$ and $\tau_{\text{OUT}}^{\text{lp}}$, respectively.

Consider the scheduling window $[0, T_{\text{OUT}}^{\text{lp}})$ of the first job of $\tau_{\text{OUT}}^{\text{lp}}$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$. We will in turn look closer at the three subintervals $[0, \Delta(k - D_{\text{IN}}^{\text{lp}}))$, $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$ and $[\Delta k, T_{\text{OUT}}^{\text{lp}})$.

We start by considering the first interval $[0, \Delta(k - D_{\text{IN}}^{\text{lp}}))$. From Lemma III.8 we know that the tasks in $\mathcal{T}_{\text{IN}}^{\Delta}$ execute for a total duration of $2\Delta^2 \text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}})$ during this interval. By Lemma III.15, the tasks in $\mathcal{T}_{\text{FIX}}^{k}$ execute for a total duration of $k(\Delta - C_{\text{FIX}}^{\text{lp}})$ in the same interval. Let x be the total duration that the processor is available for executing $\tau_{\text{OUT}}^{\text{lp}}$ during interval $[0, \Delta(k - D_{\text{IN}}^{\text{lp}}))$. We must then have

$$x = \Delta(k - D_{\text{IN}}^{\text{lp}}) - k(\Delta - C_{\text{FIX}}^{\text{lp}}) - 2\Delta^2 \text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}) \quad (45)$$

$$= kC_{\text{FIX}}^{\text{lp}} - \Delta D_{\text{IN}}^{\text{lp}} - 2\Delta^2 \text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}). \quad (46)$$

Using Eq. (10) we see that the total remaining execution time of the first job from $\tau_{\text{OUT}}^{\text{lp}}$ at time point $\Delta(k - D_{\text{IN}}^{\text{lp}})$ must be

$$C_{\text{OUT}}^{\text{lp}} - x \;=\; \Delta C_{\text{IN}}^{\text{lp}}. \quad (47)$$

Now we consider the middle subinterval $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$. We note that no job from a task in $\mathcal{T}_{\text{FIX}}^{k}$ is active in time interval $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$ by Lemma III.14. We then note that the start of the interval is a multiple of $\text{HP}(\mathcal{T}_{\text{IN}}^{\Delta})$ because

$$\Delta(k - D_{\text{IN}}^{\text{lp}}) \;=\; 2\Delta^2 \text{HP}(\mathcal{T}_{\text{IN}}) \quad (48)$$

and $\text{HP}(\mathcal{T}_{\text{IN}}^{\Delta})$ must divide $\Delta\text{HP}(\mathcal{T}_{\text{IN}})$. Since the interval is $\Delta D_{\text{IN}}^{\text{lp}}$ time units long and the start of it aligns with a hyper-period of $\mathcal{T}_{\text{IN}}^{\Delta}$, it follows that the total duration that jobs from $\mathcal{T}_{\text{IN}}^{\Delta}$ are executed in $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$ must be the same as during interval $[0, \Delta D_{\text{IN}}^{\text{lp}})$. But since $\mathcal{T}_{\text{IN}}^{\Delta}$ is just $\mathcal{T}_{\text{IN}}^{\text{hp}}$ scaled by Δ, this must be Δ times the amount of execution received by tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ during interval $[0, D_{\text{IN}}^{\text{lp}})$ in the synchronous arrival sequence of $\mathcal{T}_{\text{IN}}$. Therefore the total duration where the processor is available for $\tau_{\text{OUT}}^{\text{lp}}$ during $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$ is Δ times the total duration the processor is available for $\tau_{\text{IN}}^{\text{lp}}$ during $[0, D_{\text{IN}}^{\text{lp}})$ in the synchronous arrival sequence of $\mathcal{T}_{\text{IN}}$.

Last we consider now the subinterval $[\Delta k, T_{\text{OUT}}^{\text{lp}})$. By Lemma III.10 we know that in the synchronous arrival sequence of the constant task set $\mathcal{T}_{\text{FIX}}$, the higher-priority tasks in $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ would fully occupy the processor during $[\Delta, T_{\text{FIX}}^{\text{lp}})$. Since $\mathcal{T}_{\text{FIX}}^{k} \subseteq \mathcal{T}_{\text{OUT}}^{\text{hp}}$ and $\mathcal{T}_{\text{FIX}}^{k}$ is $\mathcal{T}_{\text{FIX}}^{\text{hp}}$ scaled by k, it follows that in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$, the higher-priority tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ fully occupies the processor during time interval $[k\Delta, kT_{\text{FIX}}^{\text{lp}}) = [\Delta k, T_{\text{OUT}}^{\text{lp}})$, and therefore $\tau_{\text{OUT}}^{\text{lp}}$ will not be able to execute after Δk.

Putting all of the above together we know that $\tau_{\text{OUT}}^{\text{lp}}$ will execute for a total of x time units during the first interval $[0, \Delta(k - D_{\text{IN}}^{\text{lp}}))$ and will not execute at all during the last interval $[\Delta k, T_{\text{OUT}}^{\text{lp}})$. To meet its deadline it must then receive a total of $C_{\text{OUT}}^{\text{lp}} - x \;=\; \Delta C_{\text{IN}}^{\text{lp}}$ time units of execution during the middle interval $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$. We know that the execution time received by $\tau_{\text{OUT}}^{\text{lp}}$ during $[\Delta(k - D_{\text{IN}}^{\text{lp}}), \Delta k)$ in the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$ is Δ times the execution time received by $\tau_{\text{IN}}^{\text{lp}}$ during $[0, D_{\text{IN}}^{\text{lp}})$ in the synchronous arrival sequence of $\mathcal{T}_{\text{IN}}$. Task $\tau_{\text{OUT}}^{\text{lp}}$ will therefore receive the remaining amount $\Delta C_{\text{IN}}^{\text{lp}}$ if and only if $\tau_{\text{IN}}^{\text{lp}}$ receives the amount $C_{\text{IN}}^{\text{lp}}$ that it requires during its scheduling window. We conclude that $\tau_{\text{OUT}}^{\text{lp}}$ is schedulable if and only if $\tau_{\text{IN}}^{\text{lp}}$ is schedulable. ■

The main result follows.

Theorem III.17. *Deciding whether a set of implicit-deadline sporadic or synchronous periodic tasks is FP-schedulable with RM priority ordering on a single preemptive processor is (weakly) NP-hard, even when restricted to task sets with utilization bounded from above by any constant $c > \ln(2)$.*

Proof: Given any constant $c > \ln(2)$, we have described how to create a reduction from an NP-hard problem (from Theorem II.3) to the target problem. This is a correct many-one reduction by Lemmas III.2–III.4 and III.16. From Eqs. (6)–(12) it is clear that the transformation required by the reduction can be computed in polynomial time. ■

IV. Hardness with Arbitrary Priorities

In this section we will see that if we may specify some particular non-RM priority ordering, then the FP-schedulability problem for implicit-deadline tasks is (weakly) NP-hard even if we restrict it to task sets with utilization bounded by any constant $c > 0$.

Similar to the case with RM priorities in the previous section, we will create a family of reductions from the FP-schedulability problem that is described in Theorem II.3. This case is more straightforward, however.

A. The reduction

Let the constant $c > 0$ be fixed and assume, without loss of generality, that $c \leqslant 1$. The source problem for the reduction is then the decision problem of Theorem II.3 where utilization is bounded by

$$c_{\text{IN}} \stackrel{\text{def}}{=} \frac{c}{2}. \quad (49)$$

Let $\mathcal{T}_{\text{IN}}$ be any given instance of the source problem. We use the same notation as in the previous section:

- $\tau_{\text{IN}}^{\text{lp}}$ is the lowest-priority task in $\mathcal{T}_{\text{IN}}$ (under RM),
- $C_{\text{IN}}^{\text{lp}}$, $D_{\text{IN}}^{\text{lp}}$ and $T_{\text{IN}}^{\text{lp}}$ are the worst-case execution time, relative deadline and period of task $\tau_{\text{IN}}^{\text{lp}}$, respectively, and
- $\mathcal{T}_{\text{IN}}^{\text{hp}}$ is the set of all higher-priority tasks in $\mathcal{T}_{\text{IN}}$.

Let $\mathcal{T}_{\text{OUT}}$ be the task set produced by the reduction. The higher-priority tasks in $\mathcal{T}_{\text{IN}}$ will be included in $\mathcal{T}_{\text{OUT}}$ without modification:

$$\mathcal{T}_{\text{OUT}}^{\text{hp}} \stackrel{\text{def}}{=} \mathcal{T}_{\text{IN}}^{\text{hp}}. \tag{50}$$

The lowest-priority task $\mathcal{T}_{\text{IN}}$ will be included with modified parameters. Let $\tau_{\text{OUT}}^{\text{lp}}$ denote this task. Its execution time $C_{\text{OUT}}^{\text{lp}}$ is copied directly from $\tau_{\text{IN}}^{\text{lp}}$:

$$C_{\text{OUT}}^{\text{lp}} \stackrel{\text{def}}{=} C_{\text{IN}}^{\text{lp}} \tag{51}$$

Its deadline and period are changed:

$$D_{\text{OUT}}^{\text{lp}} \stackrel{\text{def}}{=} T_{\text{OUT}}^{\text{lp}} \stackrel{\text{def}}{=} \text{HP}(\mathcal{T}_{\text{IN}}) + D_{\text{IN}}^{\text{lp}} \tag{52}$$

Finally, one additional task $\tau_{\text{OUT}}^{\text{fill}}$ is included in $\mathcal{T}_{\text{OUT}}$. The task $\tau_{\text{OUT}}^{\text{fill}}$ has execution time

$$C_{\text{OUT}}^{\text{fill}} \stackrel{\text{def}}{=} \text{HP}(\mathcal{T}_{\text{IN}})(1 - \text{U}(\mathcal{T}_{\text{OUT}}^{\text{hp}})) \tag{53}$$

and deadline and period

$$D_{\text{OUT}}^{\text{fill}} \stackrel{\text{def}}{=} T_{\text{OUT}}^{\text{fill}} \stackrel{\text{def}}{=} \left\lceil \frac{4C_{\text{OUT}}^{\text{fill}}}{c} \right\rceil. \tag{54}$$

The produced task set is then

$$\mathcal{T}_{\text{OUT}} \stackrel{\text{def}}{=} \mathcal{T}_{\text{OUT}}^{\text{hp}} \cup \{\tau_{\text{OUT}}^{\text{fill}}, \tau_{\text{OUT}}^{\text{lp}}\}. \tag{55}$$

Definition IV.1 (The priority ordering)**.** *The produced task set $\mathcal{T}_{\text{OUT}}$ is given the following priority ordering: The tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ have the highest priorities in $\mathcal{T}_{\text{OUT}}$, with their internal priority ordering being RM. The task $\tau_{\text{OUT}}^{\text{fill}}$ has the next-to-lowest priority, and the task $\tau_{\text{OUT}}^{\text{lp}}$ has the lowest priority.*

We note that the priority ordering specified above is simply the RM priority ordering where the priorities of the two lowest-priority tasks have been switched.

B. Correctness of the reduction

Here we show that the reduction described above is a correct many-one reduction. The idea is that $\tau_{\text{OUT}}^{\text{fill}}$ fills up all the remaining processor time left over from the higher-priority tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ during interval $[0, \text{HP}(\mathcal{T}_{\text{IN}}))$, and is then inactive during interval $[\text{HP}(\mathcal{T}_{\text{IN}}), D_{\text{OUT}}^{\text{lp}}) = [\text{HP}(\mathcal{T}_{\text{IN}}), \text{HP}(\mathcal{T}_{\text{IN}}) + D_{\text{IN}}^{\text{lp}})$. The first job of the lowest-priority task $\tau_{\text{OUT}}^{\text{lp}}$ can then only execute during interval $[\text{HP}(\mathcal{T}_{\text{IN}}), \text{HP}(\mathcal{T}_{\text{IN}}) + D_{\text{IN}}^{\text{lp}})$ where it will face the same interference as $\tau_{\text{IN}}^{\text{lp}}$ faces during $[0, D_{\text{IN}}^{\text{lp}})$. See Figure 3 for an illustration.

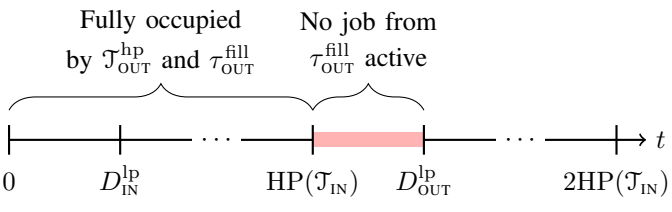

Fig. 3. The first job from $\tau_{\text{OUT}}^{\text{lp}}$ will be blocked from executing before $\text{HP}(\mathcal{T}_{\text{IN}})$ and will then experience exactly the same interference during $[\text{HP}(\mathcal{T}_{\text{IN}}), D_{\text{OUT}}^{\text{lp}})$ as $\tau_{\text{IN}}^{\text{lp}}$ would experience during $[0, D_{\text{IN}}^{\text{lp}})$.

We begin by noting that $\mathcal{T}_{\text{OUT}}$ is a valid instance of the target problem.

Lemma IV.2. *$\mathcal{T}_{\text{OUT}}$ has implicit deadlines, positive integer task parameters and $\text{U}(\mathcal{T}_{\text{OUT}}) \leqslant c$.*

Proof: The tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ have implicit deadlines since the tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ have so by definition. The tasks $\tau_{\text{OUT}}^{\text{fill}}$ and $\tau_{\text{OUT}}^{\text{lp}}$ have implicit deadlines by construction.

We note that $C_{\text{OUT}}^{\text{fill}}$ is a positive integer by Eq. (53) because $(1 - \text{U}(\mathcal{T}_{\text{OUT}}^{\text{hp}})) > 0$ and $\text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{OUT}}^{\text{hp}}) = \text{HP}(\mathcal{T}_{\text{IN}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}})$, which is a multiple of the integer $\text{HP}(\mathcal{T}_{\text{IN}}^{\text{hp}})\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}})$. All other task parameters are easily seen to be positive integers as well.

By Eqs. (51) and (52) we have $C_{\text{OUT}}^{\text{lp}} = C_{\text{IN}}^{\text{lp}}$ and $T_{\text{OUT}}^{\text{lp}} > T_{\text{IN}}^{\text{lp}}$, so we must have $\text{U}(\tau_{\text{OUT}}^{\text{lp}}) < \text{U}(\tau_{\text{IN}}^{\text{lp}})$. It follows that

$$\begin{aligned} \text{U}(\mathcal{T}_{\text{OUT}}^{\text{hp}}) + \text{U}(\tau_{\text{OUT}}^{\text{lp}}) &< \text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}) + \text{U}(\tau_{\text{IN}}^{\text{lp}}) && (56)\\ &= \text{U}(\mathcal{T}_{\text{IN}}) && (57)\\ &\leqslant c_{\text{IN}} && (58)\\ &= \frac{c}{2}. && (59) \end{aligned}$$

We also have

$$\begin{aligned} \text{U}(\tau_{\text{OUT}}^{\text{fill}}) &= \frac{C_{\text{OUT}}^{\text{fill}}}{T_{\text{OUT}}^{\text{fill}}} && (60)\\ &= \frac{C_{\text{OUT}}^{\text{fill}}}{\lceil 4C_{\text{OUT}}^{\text{fill}}/c \rceil} && (61)\\ &\leqslant \frac{c}{4}, && (62) \end{aligned}$$

and hence $\text{U}(\mathcal{T}_{\text{OUT}}) = \text{U}(\mathcal{T}_{\text{OUT}}^{\text{hp}}) + \text{U}(\tau_{\text{OUT}}^{\text{lp}}) + \text{U}(\tau_{\text{OUT}}^{\text{fill}}) < c$. ■

Next we note that the tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ and $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ must be schedulable.

Lemma IV.3. *If $\mathcal{T}_{\text{IN}}$ is FP-scheduled with RM priorities, then the higher-priority tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ are schedulable. If $\mathcal{T}_{\text{OUT}}$ is FP-scheduled with the priority ordering specified in Definition IV.1, then the higher-priority tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ are schedulable.*

Proof: We have $\text{U}(\mathcal{T}_{\text{IN}}^{\text{hp}}) < \text{U}(\mathcal{T}_{\text{IN}}) \leqslant c/2 \leqslant 1/2$. Since the tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$ have implicit deadlines they are schedulable with RM priorities by Theorem II.1.

The tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ are identical to the tasks in $\mathcal{T}_{\text{IN}}^{\text{hp}}$. Since they have RM priorities internally and also higher priorities than the other tasks in $\mathcal{T}_{\text{OUT}}$, they must also be schedulable. ■

We then note that the interference from higher-priority tasks upon $\tau_{\text{OUT}}^{\text{lp}}$ is easily characterized.

Lemma IV.4. *In the synchronous arrival sequence of $\mathcal{T}_{\text{OUT}}$, the processor will be completely busy with executing jobs from tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}} \cup \{\tau_{\text{OUT}}^{\text{fill}}\}$ in time interval $[0, \text{HP}(\mathcal{T}_{\text{IN}}))$, and all jobs released by those tasks inside the interval will have finished by time point $\text{HP}(\mathcal{T}_{\text{IN}})$.*

Proof: We have $\text{HP}(\mathcal{T}_{\text{OUT}}^{\text{hp}}) = \text{HP}(\mathcal{T}_{\text{IN}}^{\text{hp}})$ and therefore $\text{HP}(\mathcal{T}_{\text{IN}})$ is a multiple of $\text{HP}(\mathcal{T}_{\text{OUT}}^{\text{hp}})$. The tasks in $\mathcal{T}_{\text{OUT}}^{\text{hp}}$ have implicit deadlines and must therefore all have a deadline at

time point $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$. By Lemma IV.3 the tasks in $\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}$ are schedulable, and so all jobs from tasks in $\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}$ released before time point $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$ must have finished by $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$. The total execution time of those jobs is $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})\mathrm{U}(\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}})$.

The processor is then *not* busy executing jobs from $\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}$ for a total duration of $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})(1-\mathrm{U}(\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}))$ during $[0, \mathrm{HP}(\mathcal{T}_{\mathrm{IN}}))$. This is exactly the same as $C^{\mathrm{fill}}_{\mathrm{OUT}}$, so it follows that the first job from $\tau^{\mathrm{fill}}_{\mathrm{OUT}}$ finish no later than $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$ and that the processor is completely busy executing jobs from $\mathcal{T}_{\mathrm{OUT}} \cup \{\tau^{\mathrm{fill}}_{\mathrm{OUT}}\}$ during time interval $[0, \mathrm{HP}(\mathcal{T}_{\mathrm{IN}}))$. ■

We can now show that the reduction is correct.

Lemma IV.5. *Task set $\mathcal{T}_{\mathrm{OUT}}$ is FP-schedulable with the priority ordering in Definition IV.1 if and only if $\mathcal{T}_{\mathrm{IN}}$ is FP-schedulable with RM priority ordering.*

Proof: From Lemma IV.3 we know that $\mathcal{T}_{\mathrm{IN}}$ is schedulable if and only if its lowest-priority task $\tau^{\mathrm{lp}}_{\mathrm{IN}}$ is schedulable, and that $\mathcal{T}_{\mathrm{OUT}}$ is schedulable if and only if $\tau^{\mathrm{fill}}_{\mathrm{OUT}}$ and $\tau^{\mathrm{lp}}_{\mathrm{OUT}}$ are.

Since $c \leqslant 1$ and $\mathrm{U}(\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}) = \mathrm{U}(\mathcal{T}^{\mathrm{hp}}_{\mathrm{IN}}) < \mathrm{U}(\mathcal{T}_{\mathrm{IN}}) \leqslant c_{\mathrm{IN}} = c/2$ we have

$$T^{\mathrm{fill}}_{\mathrm{OUT}} = \left\lceil \frac{4C^{\mathrm{fill}}_{\mathrm{OUT}}}{c} \right\rceil \tag{63}$$

$$= \left\lceil \frac{4\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})(1-\mathrm{U}(\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}))}{c} \right\rceil \tag{64}$$

$$\geqslant \left\lceil \frac{4\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})(1-(c/2))}{c} \right\rceil \tag{65}$$

$$= \left\lceil \left(\frac{4}{c} - 2\right)\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}) \right\rceil \tag{66}$$

$$\geqslant 2\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}). \tag{67}$$

From Lemma IV.4 we know that the first job from $\tau^{\mathrm{fill}}_{\mathrm{OUT}}$ finish no later than time point $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$ in the synchronous arrival sequence of $\mathcal{T}_{\mathrm{OUT}}$. Because $D^{\mathrm{fill}}_{\mathrm{OUT}} = T^{\mathrm{fill}}_{\mathrm{OUT}} \geqslant 2\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$ it follows that task $\tau^{\mathrm{fill}}_{\mathrm{OUT}}$ must be schedulable. Since $T^{\mathrm{fill}}_{\mathrm{OUT}} \geqslant 2\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}) \geqslant \mathrm{HP}(\mathcal{T}_{\mathrm{IN}}) + D^{\mathrm{lp}}_{\mathrm{IN}} = D^{\mathrm{lp}}_{\mathrm{OUT}}$, it also follows that the second job from $\tau^{\mathrm{fill}}_{\mathrm{OUT}}$ is released after $D^{\mathrm{lp}}_{\mathrm{OUT}}$ and hence that no job from $\tau^{\mathrm{fill}}_{\mathrm{OUT}}$ is active during interval $[\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}), D^{\mathrm{lp}}_{\mathrm{OUT}})$ in the synchronous arrival sequence of $\mathcal{T}_{\mathrm{OUT}}$.

From Lemma IV.4 we know that task $\tau^{\mathrm{lp}}_{\mathrm{OUT}}$ has not executed at all during interval $[0, \mathrm{HP}(\mathcal{T}_{\mathrm{IN}}))$ in the synchronous arrival sequence. The task $\tau^{\mathrm{lp}}_{\mathrm{OUT}}$, and therefore the whole task set $\mathcal{T}_{\mathrm{OUT}}$, is then schedulable if and only if the first job of $\tau^{\mathrm{lp}}_{\mathrm{OUT}}$ receives $C^{\mathrm{lp}}_{\mathrm{OUT}}$ units of execution time during the remainder of its scheduling window, which is the time interval $[\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}), D^{\mathrm{lp}}_{\mathrm{OUT}}) = [\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}), \mathrm{HP}(\mathcal{T}_{\mathrm{IN}}) + D^{\mathrm{lp}}_{\mathrm{IN}})$.

Because $\mathrm{HP}(\mathcal{T}_{\mathrm{IN}})$ is a multiple of $\mathrm{HP}(\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}})$, the amount of processor time left over by the tasks in $\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}$ during $[\mathrm{HP}(\mathcal{T}_{\mathrm{IN}}), \mathrm{HP}(\mathcal{T}_{\mathrm{IN}}) + D^{\mathrm{lp}}_{\mathrm{IN}})$ is the same as during interval $[0, D^{\mathrm{lp}}_{\mathrm{IN}})$. The task set $\mathcal{T}_{\mathrm{OUT}}$ is then schedulable if and only if at least $C^{\mathrm{lp}}_{\mathrm{OUT}}$ time units of processor time is left over from $\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}}$ during $[0, D^{\mathrm{lp}}_{\mathrm{IN}})$, but since $\mathcal{T}^{\mathrm{hp}}_{\mathrm{OUT}} = \mathcal{T}^{\mathrm{hp}}_{\mathrm{IN}}$ and $C^{\mathrm{lp}}_{\mathrm{OUT}} = C^{\mathrm{lp}}_{\mathrm{IN}}$, this is equivalent to saying that the first job of task $\tau^{\mathrm{lp}}_{\mathrm{IN}}$ receives $C^{\mathrm{lp}}_{\mathrm{IN}}$ units of execution time during its scheduling window. We conclude that $\mathcal{T}_{\mathrm{OUT}}$ is schedulable if and only if $\mathcal{T}_{\mathrm{IN}}$ is. ■

The hardness of the target problem follows.

Theorem IV.6. *Deciding whether a set of implicit-deadline sporadic or synchronous periodic tasks is FP-schedulable on a single preemptive processor when any priority ordering can be given is (weakly) NP-hard, even when restricted to task sets with utilization bounded from above by any constant $c > 0$.*

Proof: We have described a reduction from an NP-hard problem (from Theorem II.3) to the target problem, for any given constant $c > 0$. It is easy to compute the reduction in polynomial time following Eqs. (50)–(55), and by Lemmas IV.2 and IV.5 it is a correct many-one reduction. ■

V. Conclusions

Thanks to Liu and Layland's utilization bound it is trivial to conclude that a task set with implicit deadlines and utilization no larger than $\ln(2)$ is FP-schedulable with RM priorities. We have seen in this paper that if we allow task sets with utilization up to some constant $c > \ln(2)$, then deciding if a task set is FP-schedulable with RM priorities becomes weakly NP-complete, even if c exceeds $\ln(2)$ by just an arbitrarily small amount. This again highlights the importance of $\ln(2)$ as a transition point where the FP-schedulability problem with RM priorities undergoes a transformation. From surely schedulable to possibly unschedulable, as we know from Liu and Layland, but also from computationally easy to computationally hard.

We have also seen that if we can specify an arbitrary priority ordering to the FP-schedulability problem for implicit-deadline tasks, then it is weakly NP-complete already if we bound the utilization of considered task sets by any constant $c > 0$.

References

[1] C.-L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[2] O. Serlin, "Scheduling of time critical processes," in *Proceedings of the 1972 Spring Joint Computer Conference*, p. 925–932.

[3] E. Bini, G. C. Buttazzo, and G. M. Buttazzo, "Rate monotonic analysis: the hyperbolic bound," *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 933–942, 2003.

[4] M. Joseph and P. Pandya, "Finding response times in a real-time system," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 1986.

[5] P. Ekberg and W. Yi, "Fixed-priority schedulability of sporadic tasks on uniprocessors is NP-hard," in *Proceedings of the 38th Real-Time Systems Symposium (RTSS)*, 2017, pp. 139–146.

[6] F. Eisenbrand and T. Rothvoß, "Static-priority real-time scheduling: Response time computation is NP-hard," in *Proceedings of the 29th Real-Time Systems Symposium (RTSS)*, 2008, pp. 397–406.

[7] T. Rothvoß, "On the computational complexity of periodic scheduling," Ph.D. dissertation, SB, Lausanne, 2009.

[8] S. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *Proceedings of the 11th Real-Time Systems Symposium (RTSS)*, 1990, pp. 182–190.

[9] P. Ekberg and W. Yi, "Uniprocessor feasibility of sporadic tasks with constrained deadlines is strongly coNP-complete," in *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS)*, 2015, pp. 281 – 286.

[10] ——, "Uniprocessor feasibility of sporadic tasks remains coNP-complete under bounded utilization," in *Proceedings of the 36th Real-Time Systems Symposium (RTSS)*, 2015, pp. 87–95.

[11] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proceedings of the 11th Real-Time Systems Symposium (RTSS)*, Dec 1990, pp. 201–209.
[12] R. Devillers and J. Goossens, "Liu and Layland's schedulability test revisited," *Information Processing Letters*, vol. 73, no. 5, pp. 157 – 161, 2000.

⊠